PUC Minas

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Laboratório de Algoritmos e Estruturas de Dados – 2/2023

AULA PRÁTICA – ALGORITMOS DE ORDENAÇÃO – SHELL – MERGE E QUICK

Prof. Edwaldo Soares Rodrigues

Uma discussão muito comum acontece com relação à escolha de um algoritmo de ordenação. Os algoritmos de ordenação podem ser subdivididos em duas categorias: simples (menos eficientes) e eficientes.

O objetivo desta aula prática consiste em comparar o desempenho dos seguintes algoritmos de ordenação (ShellSort, MergeSort e QuickSort) e testá-los com diversos vetores de entrada (vetores de números inteiros), contabilizando o número de comparações de chaves, o número de movimentações de registros e o tempo de execução. Para isso você deverá colocar contadores em seu código e instrumentá-lo de forma a obter o tempo de execução (use o namespace <System.Diagnostics>. Utilize uma instância da classe Stopwatch, para conseguir medir o tempo gasto pelos algoritmos. (Pesquise o funcionamento da classe Stopwatch).

Além do código, crie um documento contendo tabelas comparando a performance de cada algoritmo. Mais especificamente, você deverá realizar testes com vetores de tamanhos 1000, 10000, 100000 e 200000 elementos. Três diferentes tipos de vetores devem ser utilizados: aleatórios, ordenados e inversamente ordenados. Para facilitar a geração desse documento, salve em um arquivo de texto o número de comparações, movimentações e o tempo gasto, para cada um dos métodos e arrays a serem ordenados.

O tempo de execução pode variar de uma execução para outra, isso porque fatores como: hardware e ambiente de execução podem influenciar. Neste contexto, repita os testes pelo menos 10 vezes, de forma a obter médias do tempo de execução e dos contadores. O próprio programa deve gerar os vetores dos quatro tipos de entradas.

O seu programa deverá imprimir o método utilizado, o tipo e tamanho do vetor, o tempo de execução e o número de comparações e movimentações efetuado. Deverá haver também uma opção no programa para imprimir os vetores antes e depois da execução.

Repita os testes anteriores, só que agora para arrays cujos elementos são dados de estudantes (classe estudante, contendo os seguintes atributos: matrícula(int), nome(string), curso matriculado(string), coeficiente de rendimento(int), período atual(int). Utilize a classe Random para gerar um número aleatório, assim, conseguirá preencher os campos inteiros. Crie um método para conseguir gerar strings aleatórias para o nome do estudante e o curso (não precisam fazer sentido os nomes). Pesquise sobre o funcionamento da classe Random().

Ao final, compare o tempo de ordenação entre os vetores de tipos diferentes. No documento de texto a ser entregue deverão ter tabelas mostrando o tempo de execução de cada um dos métodos, para cada variação dos vetores, além de comparar os vetores de tipos diferentes também.